**进度汇总**

**2020-02-10**

目录

[一、上周进度 2](#_Toc32172444)

[二、本周计划 6](#_Toc32172445)

# 一、上周进度

1. **A\* 算法优化测试**
2. 优化方案
3. 优化方案R1: 序列元素合并

在待修复序列中存在两元素a和b，满足以下三个条件，则可将a和b两元素进行合并处理：

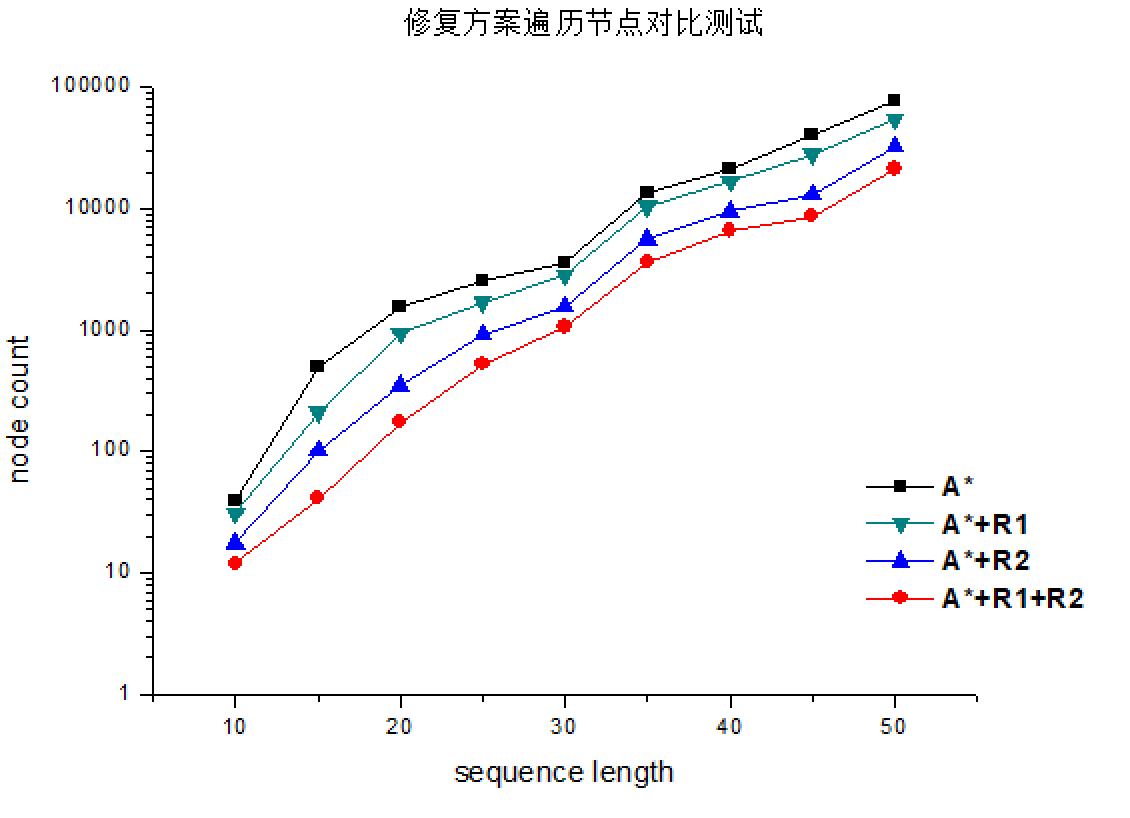
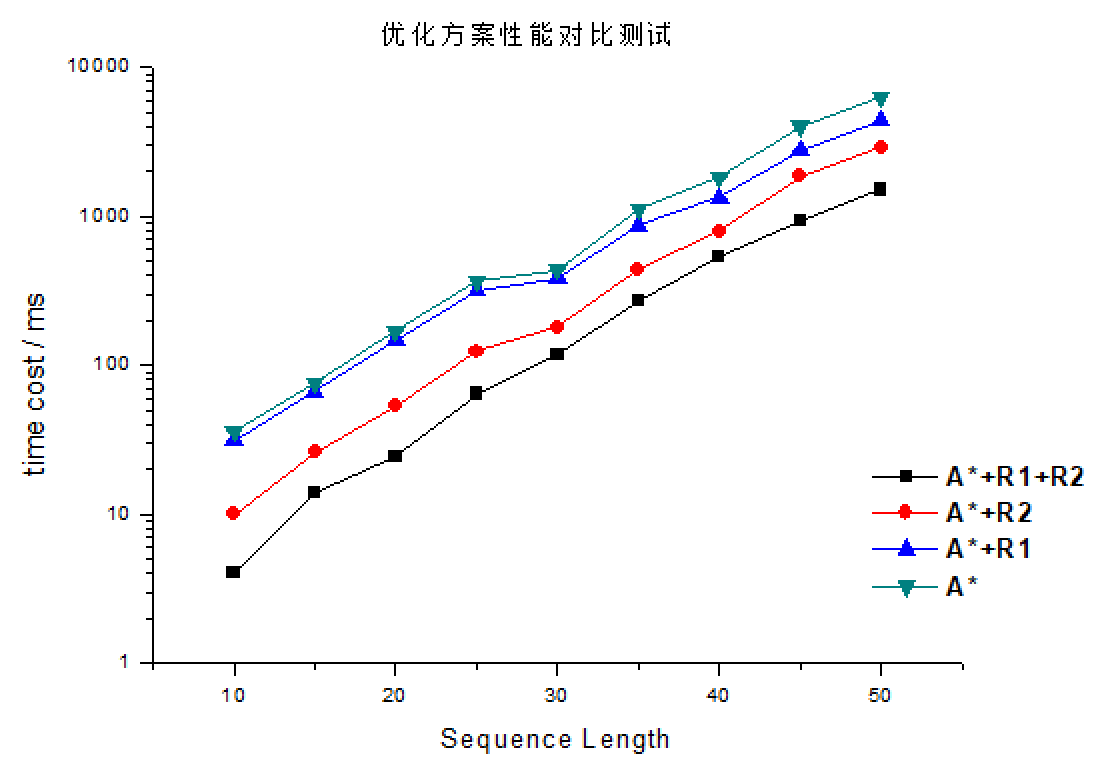
1. a和b在原序列中是连续的。
2. a和b在模型中中对应的元素也是连续的。
3. 不存在一个元素c，插入a和b之间，且保证序列是正确的。

满足上述三个条件即可将多个元素进行两两合并处理，在A\* 遍历过程中 视为一个元素处理。

1. 优化方案R2：中间状态剪枝

在A\* 搜索的过程中，每一个中间结果称为中间状态，即未完成状态，每一个中间状态都表示唯一的序列元素顺序，在遍历的过程中，将中间状态相同，修复代价高的分支进行剪枝处理。

1. 对比结果
2. 序列长度对修复算法影响

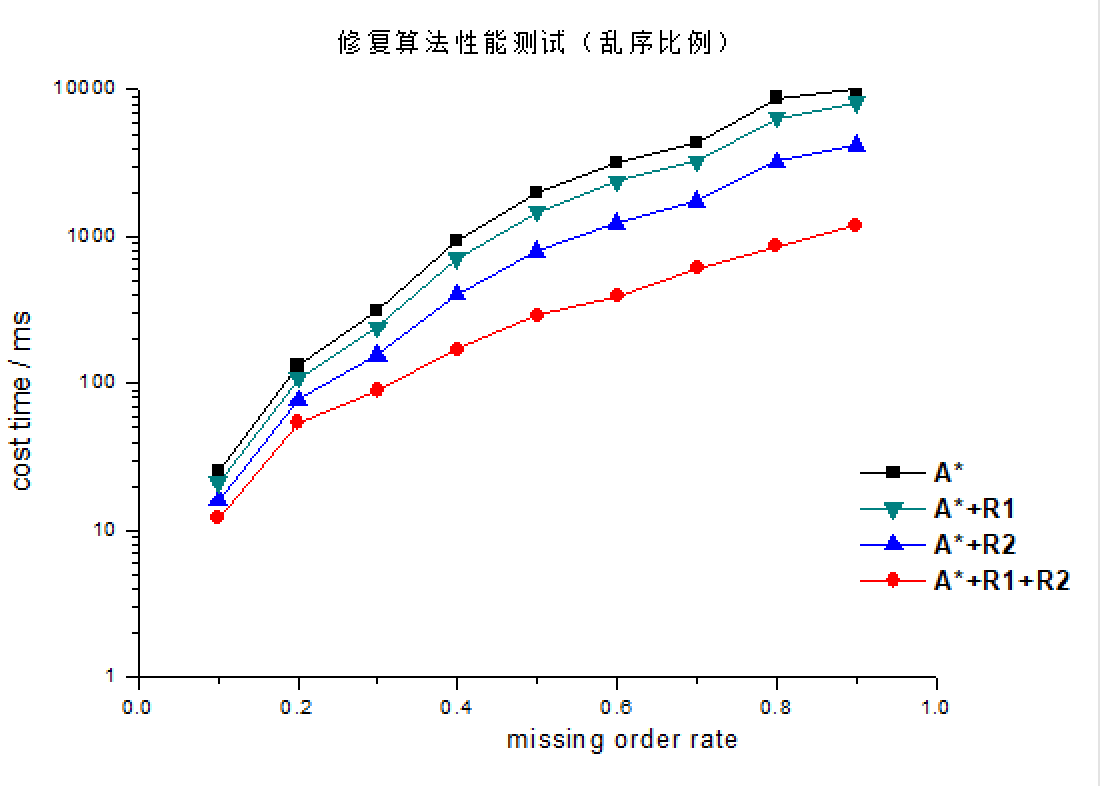


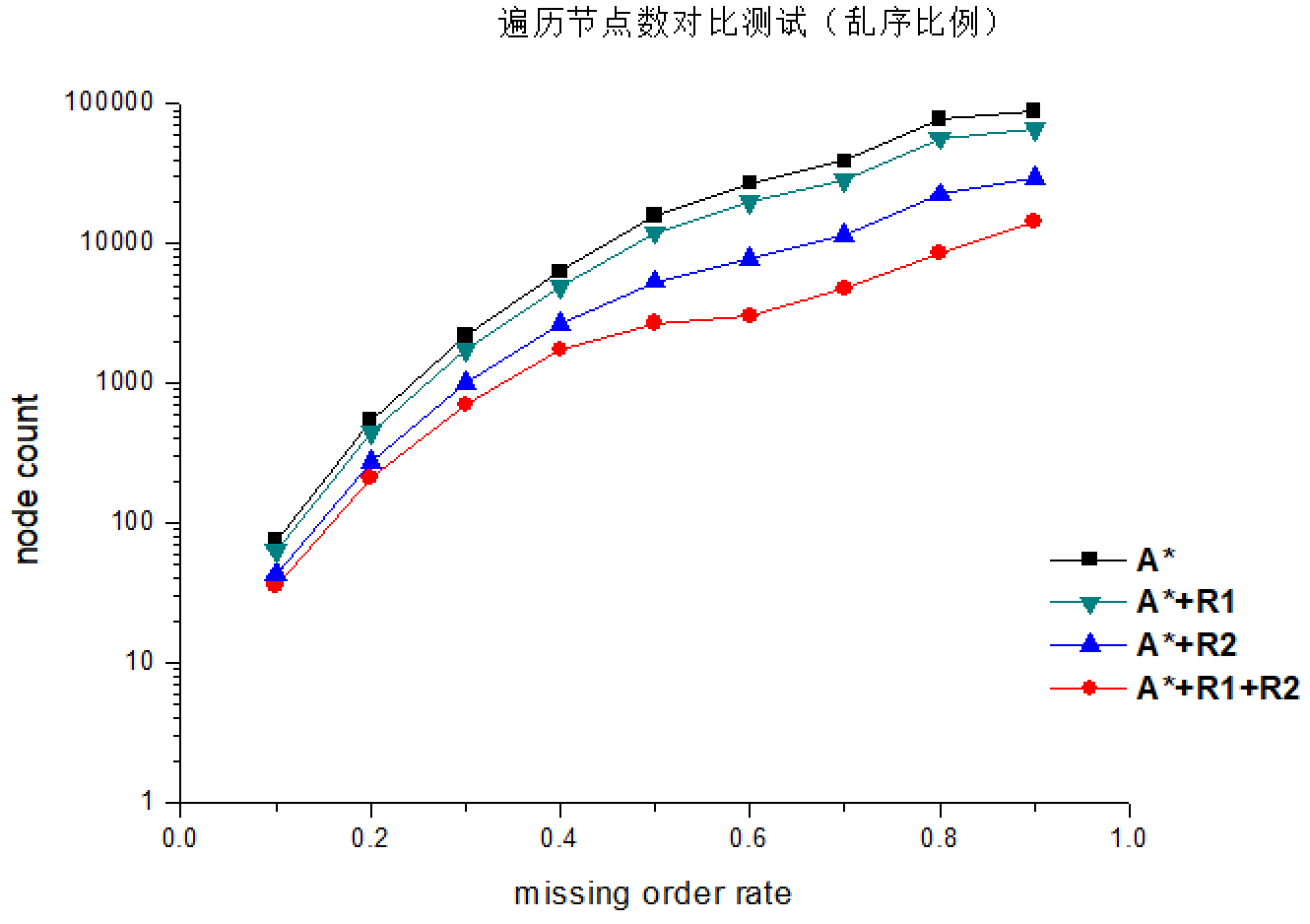
说明：

**测试序列乱序元素比例设置**：按照以下比例为每个长度的的序列生成1000个样例，最终修复时间为所有修复时间的平均值。

|  |  |  |
| --- | --- | --- |
| 序号 | 乱序元素比例随机范围 | 占比 |
| 1 | 0%～30% | 50% |
| 2 | 30%～60% | 30% |
| 3 | 60%～90% | 20% |

1. 序列乱序比例对修复算法性能影响





说明：

**测试序列长度比例设置**：按照以下比例为每个乱序比例的测试项生成1000个测试用例，最终修复时间为所有修复时间的平均值。

|  |  |  |
| --- | --- | --- |
| 序号 | 序列长度随机范围 | 占比 |
| 1 | 1-30 | 50% |
| 2 | 30-40 | 30% |
| 3 | 40-50 | 20% |

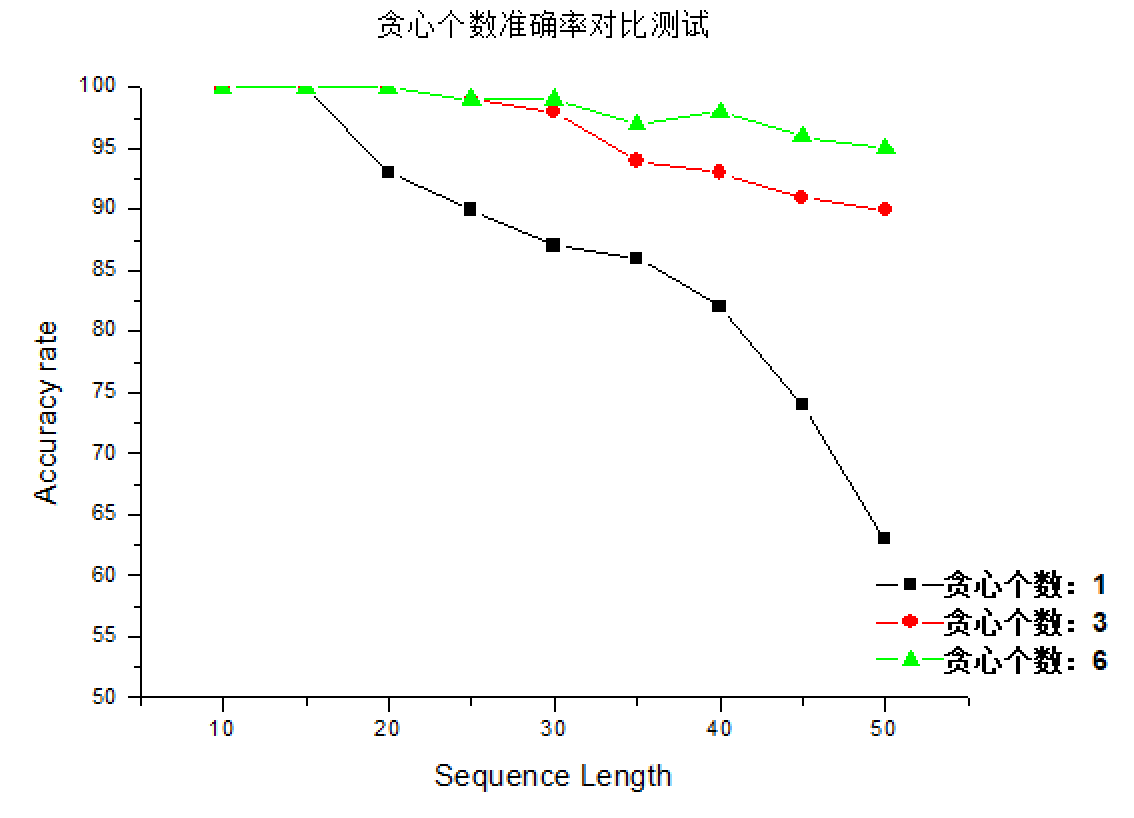
1. **最小预估代价（贪心）修复准确率提升测试**

（1） 方案

最小预估代价修复方法，采用贪心的方式，每次选取预估代价最小的元素作为基准元素，修复错误序列，直至序列中无乱序元素为止，在以往的测试中，每次贪心选取预估代价最小的元素，准确率较低。现将每次贪心元素的数量更改为一个变量，每次修复之前可以手动进行调整，以此提升贪心的准率率。

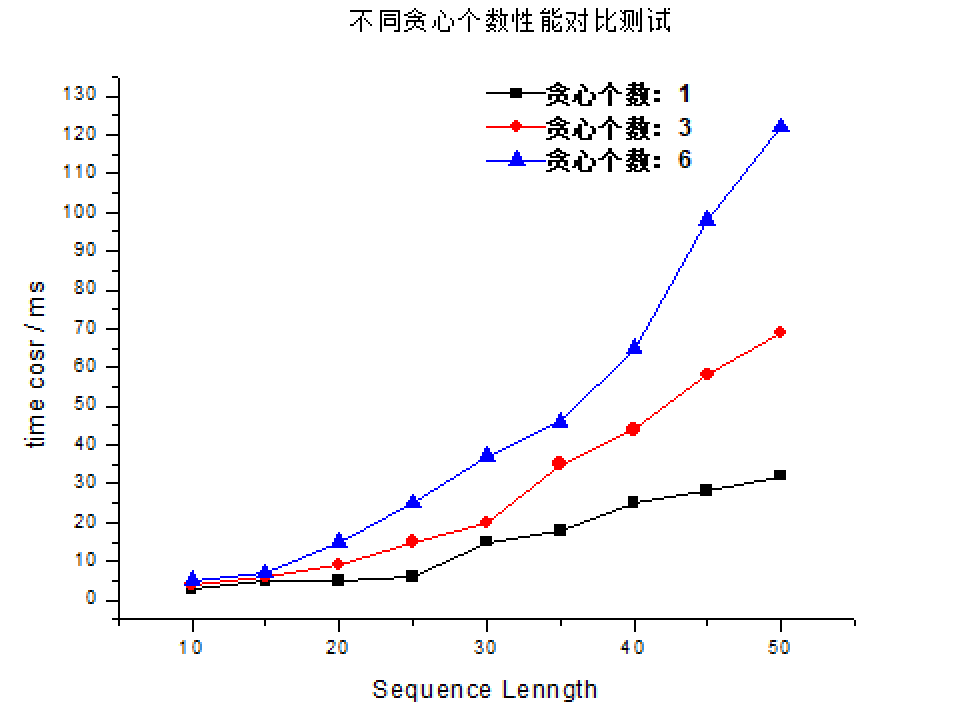
（2） 实验结果

实验结果如下图所示：



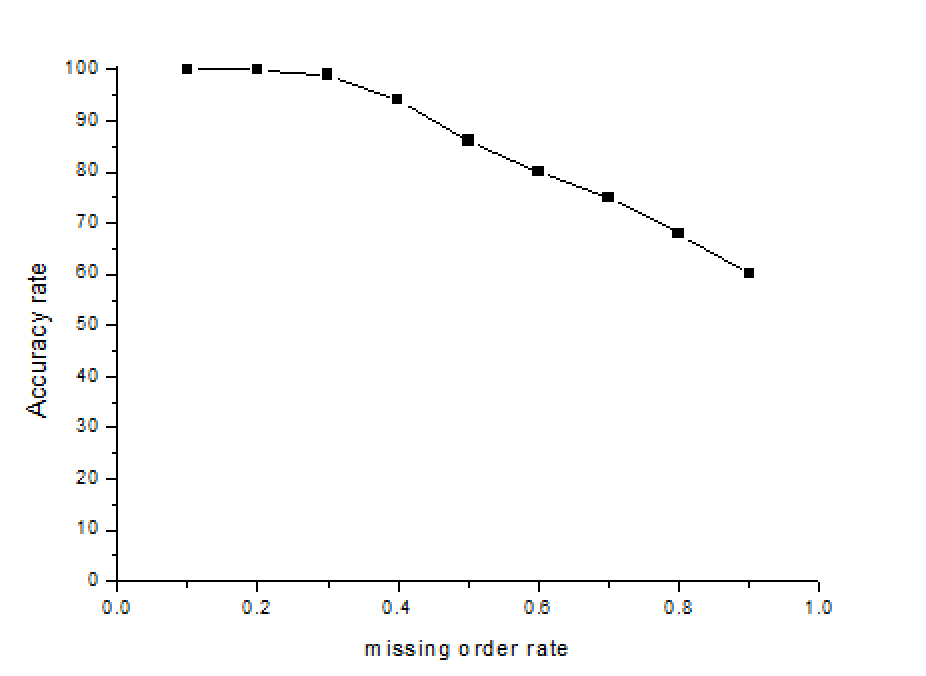
说明：上图为修复算法贪心数量为1、3和6时的准确率测试结果展示，Y轴：准确率，X轴：修复序列长度， 修复序列固定30% 的乱序元素比例。

下图为不同贪心元素个数对修复算法的性能影响，如下图所示：



**说明**：上图为修复算法贪心数量为1、3和6时的性能测试结果展示，Y轴：修复时间，X轴：修复序列长度， 修复序列固定30% 的乱序元素比例。

除了贪心个数和序列长度会影响修复准确率之外，待修复序列的乱序元素比例也是重要因素之一，测试实验结果如下：



说明：待修复序列长度固定为30.

（3） 结论

在最小预估代价修复算法中，增加每次序列调整的贪心元素个数，可以提高序列修复的准确性，但同时也会带来性能的下降，但相对于其他两种方法的性能对比，可以忽略这部分的性能损耗。

最小预估代价修复算法受序列乱序比例影响较大，因此，该算法适用于序列中存在乱序元素比例较少的场景中，避免引入过大误差。

# 二、本周计划

1. 使用永博师兄的数据集，对三种方法进行测试，目前存在个别模型无法识别修复bug，在修改中。
2. 提升循环结构处理性能，并丰富相应的测试